Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа по курсу**

**«Операционные системы»**

Студент: Копылов Михаил Юрьевич

Группа: М8О-201Б-21

Вариант: 27

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Исходный код
5. Выводы

**Репозиторий**

<https://github.com/Mikhail-cWc/OS_mai/tree/main/kp>

**Постановка задачи**

**Цель работы**

1. Приобретение практических навыков в использовании знаний, полученных в течении курса
2. Проведение исследования в выбранной предметной области

**Задание**

Необходимо написать 3-и программы. Далее будем обозначать эти программы A, B, C. Программа A принимает из стандартного потока ввода строки, а далее их отправляет программе С. Отправка строк должна производится построчно. Программа C печатает в стандартный вывод, полученную строку от программы A. После получения программа C отправляет программе А сообщение о том, что строка получена. До тех пор пока программа А не примет «сообщение о получение строки» от программы С, она не может отправлять следующую строку программе С. Программа B пишет в стандартный вывод количество отправленных символов программой А и количество принятых символов программой С. Данную информацию программа B получает от программ A и C соответственно. Способ организация межпроцессорного взаимодействия выбирает студент.

**Общие сведения о программе**

Межпроцессорное взаимодействие осуществляется с помощью пайпов и семафоров

**Исходный код**

|  |
| --- |
| **Main.cpp** |
| #include "utils.h"  int main()  {  int fdAC[2];  int fdAB[2];  int fdBC[2];  pipe(fdAC);  pipe(fdAB);  pipe(fdBC);  // Удалить именованный семафор  sem\_unlink("\_semA");  sem\_unlink("\_semB");  sem\_unlink("\_semC");  sem\_t\* semA = sem\_open("\_semA", O\_CREAT, 0777, 1);  sem\_t\* semB = sem\_open("\_semB", O\_CREAT, 0777, 0);  sem\_t\* semC = sem\_open("\_semC", O\_CREAT, 0777, 0);  if ((semA == SEM\_FAILED)||(semB == SEM\_FAILED)||(semC == SEM\_FAILED))  {  perror("sem\_open");  return -1;  }  std::cout << "Enter some strings:\n";  pid\_t C = fork();  if (C == -1)  {  perror("fork");  return -1;  }  if (C == 0)  {  pid\_t B = fork();  if (B == -1)  {  perror("fork");  return -1;  }  if (B == 0)  {  execl("B", std::to\_string(fdAB[0]).c\_str(), std::to\_string(fdAB[1]).c\_str(), std::to\_string(fdBC[0]).c\_str(), std::to\_string(fdBC[1]).c\_str(), NULL);  }  else  {  execl("C", std::to\_string(fdAC[0]).c\_str(), std::to\_string(fdAC[1]).c\_str(), std::to\_string(fdBC[0]).c\_str(), std::to\_string(fdBC[1]).c\_str(), NULL);  }  }  else  {  execl("A", std::to\_string(fdAC[0]).c\_str(), std::to\_string(fdAC[1]).c\_str(), std::to\_string(fdAB[0]).c\_str(), std::to\_string(fdAB[1]).c\_str(), NULL);  }  return 0;  } |

|  |
| --- |
| **A.cpp** |
| #include "utils.h"  int main(int args, char\* argv[])  {  int fdAC[2];  fdAC[0] = atoi(argv[0]);  fdAC[1] = atoi(argv[1]);  int fdAB[2];  fdAB[0] = atoi(argv[2]);  fdAB[1] = atoi(argv[3]);  sem\_t\* semA = sem\_open("\_semA", O\_CREAT, 0777, 1);  sem\_t\* semB = sem\_open("\_semB", O\_CREAT, 0777, 0);  sem\_t\* semC = sem\_open("\_semC", O\_CREAT, 0777, 0);  while(1)  {  std::string str;  getline(std::cin, str);    int size = str.length();  write(fdAC[1], &size, sizeof(int));  write(fdAB[1], &size, sizeof(int));  for (int i = 0; i < size; ++i)  {  write(fdAC[1], &str[i], sizeof(char));  }  sem\_post(semB);  sem\_wait(semA);  }  sem\_close(semA);  sem\_destroy(semA);  sem\_close(semB);  sem\_destroy(semB);  sem\_close(semC);  sem\_destroy(semC);  close(fdAC[0]);  close(fdAC[1]);  close(fdAB[0]);  close(fdAB[1]);  return 0;  } |

|  |
| --- |
| **B.cpp** |
| #include "utils.h"  int main(int args, char\* argv[])  {  int fdAB[2];  fdAB[0] = atoi(argv[0]);  fdAB[1] = atoi(argv[1]);  int fdBC[2];  fdBC[0] = atoi(argv[2]);  fdBC[1] = atoi(argv[3]);  sem\_t\* semA = sem\_open("\_semA", O\_CREAT, 0777, 1);  sem\_t\* semB = sem\_open("\_semB", O\_CREAT, 0777, 0);  sem\_t\* semC = sem\_open("\_semC", O\_CREAT, 0777, 0);  while (1)  {  int size;  read(fdAB[0], &size, sizeof(int));  std::cout << "Number of input symbols is " << size << std::endl;    sem\_post(semC);  sem\_wait(semB);  read(fdBC[0], &size, sizeof(int));  std::cout << "Number of output symbols is " << size << std::endl;    sem\_post(semA);  sem\_wait(semB);    }  sem\_close(semA);  sem\_close(semB);  sem\_close(semC);  close(fdAB[0]);  close(fdAB[1]);  close(fdBC[0]);  close(fdBC[1]);  return 0;  } |

|  |
| --- |
| **C.cpp** |
| #include "utils.h"  int main(int args, char\* argv[])  {  int fdAC[2];  fdAC[0] = atoi(argv[0]);  fdAC[1] = atoi(argv[1]);  int fdBC[2];  fdBC[0] = atoi(argv[2]);  fdBC[1] = atoi(argv[3]);  sem\_t\* semA = sem\_open("\_semA", O\_CREAT, 0777, 1);  sem\_t\* semB = sem\_open("\_semB", O\_CREAT, 0777, 0);  sem\_t\* semC = sem\_open("\_semC", O\_CREAT, 0777, 0);  while(1)  {  int size;  std::string str;  read(fdAC[0], &size, sizeof(int));  int t = 0;  for (int i = 0; i < size; ++i)  {  char c;  read(fdAC[0], &c, sizeof(char));  str.push\_back(c);  t = i;  }  ++t;  std::cout << str << std::endl;  write(fdBC[1], &t, sizeof(int));    sem\_post(semB);  sem\_wait(semC);  }  sem\_close(semA);  sem\_close(semB);  sem\_close(semC);  close(fdAC[0]);  close(fdAC[1]);  close(fdBC[0]);  close(fdBC[1]);  return 0;  } |

**Выводы**

Составлена и отлажена программа на языке C++, в соответствии с заданием.